NLP HW2

1. **Describe how you build your model? How did you do to preprocess your data from dataset? The distribution of the emotion is unbalance, what did you do to improve the accuracy on those emotion which are in small scale? (30%)**

   After, trying countless model architectures, I discovered that using a bi-directional LSTM with 64 neurons along with a dropout layer with parameter 0.5 has the best performance on my preprocessed data. Therefore, the whole architecture will be an embedding layer, a bi-directional LSTM layer, a dropout layer and a 7-output dense layer with softmax.

   I try many different methods of preprocessing data, including no preprocessing, only exclude special non-alpha data, only exclude stop words, only exclude stop words except "no" and "not", exclude both special non-alpha data and stop words except "no" and "not". Of all these methods, I found out that **only exclude special non-alpha data has the best performance, higher than 30% f1 score**. Surprisingly, **keeping stop words can benefit the f1 score**. I discussed this situation with several classmates, and they all face the same situation in their experiments. Surly deleting stop words would harm the performance. I think **stop words contain some vital information** might be the reason. Noticed that I did not delete all stop words. In fact, I keep "not", "no" while "deleting stop words". Furthermore, I use nltk stopwords and nltk word_tokenize to remove stop words and tokenize words. After that, I use Textvectorization from keras.layers to give each word index.

   First of all, both train and dev data have very imbalanced emotion distributions. Neutral accounts for nearly 50% of the emotions, while six other emotions share the rest. Although I cannot access the testing labels, I imagine if the testing labels have the same distribution, then balancing the training data won't achieve extra performance. Nevertheless, I still tackle the class imbalanced problem by **calculate the ratio between the max class, "neutral", and others** so that we duplicate the samples of other data by the ratio of times. Also, I implemented **data augmentation with wordnet by nlpaug library**. The amount of data in each emotion class will approximate that of the max class, "neutral". By applying the balanced data, the performance of the model increase. Note that the best performance is not the completely balanced data. Half balanced data, dividing the duplicate ratio by two, achieve the best performance in my experiments.

In conclusion, I duplicate the samples on those emotions with small scale by a ratio to make the size of each emotion approximately equal. The accuracy significantly improved to higher than 90%.

2. **Have you tried pretrain word embedding? (e.g. Glove or Word2vec).What is the influence of the result after you using them?(30%)**

   I definitely try pretrained word embedding. In fact, it is the reason why I pass the strong baseline. For many long torturing days, I couldn't improve my model's macro f1 score and couldn't find out the reason until I discover that I am using the Glove 6B pretrained model, not 840B pretrained model. It turns out 840B pretrained model can effectively identify 400 more words than the 6B one, which means the **words Glove couldn't recognize reduce from 723 to 308**. The influence of the pretrained model is powerful. My model's **f1 score** of raw data training **went from 23~26% to 26~30%** when using Glove 840B, comparing to Glove 6B.

3. **Have you tried attention on your model? What is the influence of the result after you using them? Which text your model attention on when it predicts the emotion? (30%)**

   Yes, I do. It seems like **adding attention layer** make my model **overfit faster** and does **not significantly improve the performance** of my experiments. For example, I add a attention layer and a flatten layer after bi-directional LSTM, the f1 score dropped slightly.
   By nature, attention layer should **focus on the most important words** in the context. Since the focus of attention layer is very like to align with the focus of the complete trained model. I experiment the attention of a model that achieves 30+% f1 score with several words and sentences and visualize the evaluations. For instance, int the sentence **"Why are you stupid?"**, the attention layer might place its focus on "Why". (Bold for probability > 30%)

	neutral	anger	joy	supervise	sadness	disgust	fear
Why	0.09	0.11	0.28	**0.49**	0.01	0.01	0.00
are	**0.37**	0.04	0.04	**0.55**	0.00	0.00	0.00
you	**0.37**	0.05	0.14	**0.44**	0.00	0.00	0.00
stupid	**0.64**	0.08	0.09	0.18	0.01	0.00	0.00

?	**0.62**	0.05	0.15	0.17	0.01	0.00	0.00
Sentence	0.07	0.10	0.05	**0.73**	0.03	0.02	0.01

4. **Have you used other information form dataset to improve your model performance? (e.g. Speaker) What is the influence of the result after you using them? (10%)**

   By adding the information of Speaker, Episodes into the model., the **model architecture cannot be Sequential** and has to be more flexible because they probably should not enter embedding layer. I think the performance does not visibly improve. The **f1 scores fluctuate as usual**. Thus, I do not include this information in the later experiments.